IN THE SPECIFICATION

*Please replace paragraph **[0025]** with the following amended paragraph:*

[0025] **Figure 1** illustrates one embodiment of the present invention, where the dependency chain processor consists of a control flow engine 2 and a data flow engine 7 that cooperate in producer/consumer fashion to execute programs. A control flow engine 2 reads and processes trace descriptors from the trace descriptor storage in speculative control flow order. Speculative fetching of trace descriptors is carried out by a trace fetch unit 3. Each trace descriptor 102 contains dependency chain descriptors 101 for the trace's constituent dependency chains, in addition to aggregate trace information. Upon processing by the DC (dependency chain) dispatch unit 4 of a trace descriptor, the control flow engine 2 pushes the constituent dependency chain descriptors 101 into a dependency chain issue window 6. A data flow engine 1 in turn consumes dependency chain descriptors 101 from the dependency chain issue window 6 dispatching them to available clusters 8 for execution within each execution cluster 8 instructions of the assigned dependency chain are fetched from the dependency chain instruction storage and executed. Each execution cluster 8 may include a fetch unit 11, issued unit 12 and execute unit 13.

*Please replace paragraph **[0030]** with the following amended paragraph:*

[0030] Overall, instruction fetch is a two-stage process. The fetch of traces descriptors 102 in program order is the first stage. The fetch of actual instructions of a dependency chain in dependence order is the second stage. This has two main benefits. First, the fetch and processing of trace descriptors 102 in the control flow engine 2 may allow it to step ahead faster in the dynamic ~~innstruction~~ instruction stream, because the trace descriptors 102 occupy much less space than the instructions themselves and because only aggregate resources for traces such as global reorder buffer 9 live-out entries are allocated by the control flow engine 2. This run ahead stepping through trace descriptors 102 and dispatching of dependency chain descriptors 101 can create a

large virtual issue window allowing the processor to get potentially distant yet ready to execute instructions quickly as illustrated in **Figure 5.** Fast stepping through the trace descriptors 102 by the control flow engine 2 can also allow the virtual issue window to enjoy fast refill following branch mispredictions.

*Please replace paragraph **[0055]** with the following amended paragraph:*

[0055] Another embodiment of the present invention, illustrated in **Figure 7**, is the use of the dependency chain processor 205 in a computer system with a bus 200 connecting the dependency chain processor 205 with a memory 201, storage unit 202 and other peripherals such as a keyboard 203.

*Please add the following <u>new</u> paragraph after paragraph **[0054]** :*

[0054.1] **Figure 6** is a flowchart depicting one embodiment of processing data in the dependency chain processor. A trace description may be fetched (block 601). Dependency descriptors may be separated from the trace descriptors (block 603). Dependency descriptors may be dispatched to the dependency chain issue window (block 605). The descriptors may be dispatched to execution cluster (block 607). Instructions may be fetched from storage (block 609). The instructions may be executed (block 611). The reorder buffer may be updated (block 613). The architectural state may be updated (block 615).